ON AUTOMATIC DEDUCTION

by JOHN ALAN ROBINSON

INTRODUCTION

Leibniz long ago saw that deductive reasoning and arithmetical calculation have much in common. As is well known, he even envisaged a universal logical calculus within which all problems of deduction would be formulated as problems of calculation; thus the execution of purely clerical algorithms, akin to those for addition and multiplication of numerals, would suffice to solve such problems when they were properly formulated in the calculus. The question whether a statement $\mathcal{B}$ does or does not follow logically from a set $\mathcal{A}_1,\ldots,\mathcal{A}_n$ of other statements as assumptions or premises would then be settled, not by debate and argument, but mechanically and automatically, just as for example, we settle the question whether 26,105,181 is or is not the product of 843 and 30,967. This method of solving a problem of deduction—by calculation—appears to be, and is, quite different from the usual method (if "method" is not indeed a misnomer) of making a deduction by using intelligence, intuition, insight, or, as it is often expressed, by reasoning.

The same contrast is afforded by the example of a complex intellectual game, such as chess. A player normally selects his moves in a game of chess by an arcane process of reasoning, intuitive assessment of possible positions and advantages, hunches, and the like, which he does not completely—and probably could not if challenged to do so—articulate into a routine automatic algorithm. Likewise anyone who must carry out much deductive reasoning, such as a mathematician, is unable to give a complete set of instructions stating exactly how to do it, even though his own deductive capabilities may be extremely powerful and fluent, as is usually the case among professional mathematicians.

The vision of Leibniz was not, then, a trivial one. It is by no means obvious that problems of deduction can be formulated as problems of calculation; nor, for that matter, is it obvious that they cannot. Leibniz himself was unable to settle the question, which, as we now know, requires some

---

Editor's Note: Mr. Robinson is Professor of Philosophy at Rice University.

of the most advanced theory of modern mathematical logic for its treatment. In the light of modern findings, the situation is quite clear and to some extent rather bizarre. On the one hand, Alonzo Church[1] has proved that there cannot exist a "Leibnizian" algorithm or calculation procedure for solving every problem of the form: *determine whether or not the statement $\mathcal{B}$ logically follows from the assumptions $\mathcal{A}_1,\ldots,\mathcal{A}_n$*. It is not merely that the human race will never discover such an algorithm, but that there simply is no such algorithm to be discovered or not discovered. Of course this result depends on a careful and exact analysis of the concepts involved, notably the concepts *algorithm, statement,* and *logical consequence.* At bottom such analyses are no more than definitions of these otherwise informal and loose concepts, and anyone who is not disposed to accept one or more of these definitions is not bound to accept the result of Church concerning them. Nevertheless there is wide agreement that the analyses are proper, and that consequently the nonexistence of a universal "Leibnizian" algorithm, of the sort defined, is as well established as anything can be.

In view of this negative result concerning the deduction problem, and of our introduction of the analogy of the chess-playing problem, it is of no small interest that von Neumann[2] has proved the following positive result for the game of chess: either there is an algorithm by following which the White player can always win, irrespective of how the Black player plays; or there is an algorithm by following which the Black player can always win, irrespective of how the White player plays; or there is one algorithm for each player, such that each following his own algorithm can always force a draw, irrespective of how the other player makes his moves. Von Neumann's comment on this result is worth quoting:

> But our proof, which guarantees the validity of one (and only one) of these three alternatives, gives no practically usable method to determine the true one. This relative, human difficulty necessitates the use of those incomplete, heuristic methods of playing, which constitute "good" chess; and without it there would be no element of "struggle" and "surprise" in that game.[3]

Here then the situation is theoretically totally different, for we are as mathematically certain of the existence of a chess algorithm, by von Neumann's theorem, as we are of the nonexistence of a deduction algorithm, by Church's theorem. Yet practically the situation is the same, as far as the human mind is concerned; for as von Neumann's remarks indicate, we shall never in fact possess a chess algorithm, so that the game will always be played with the help of intelligence and acumen, in a "creative" way.

It is therefore curious that we cannot draw the same conclusion about deduction that von Neumann rightly drew about chess-playing. Despite the apparent negative finality of Church's theorem, there is a positive result

which forms just as important a part of modern logical theory, which for all practical purposes would seem to vindicate Leibniz' expectations completely or nearly so, and which undoubtedly will eventually have a profound impact on the way in which deduction problems are handled. Suppose that we put the general deduction problem as follows: *deduce the statement* $\mathcal{B}$ *from the assumptions* $\mathcal{A}_1,\ldots,\mathcal{A}_n$. Each such problem is either solvable or not; and, if a solution exists in a given case, it will take the form of a *deduction,* or *proof,* of $\mathcal{B}$ from the assumptions $\mathcal{A}_1,\ldots,\mathcal{A}_n$. The somewhat surprising fact is that there exists an algorithm with which one can always calculate the solution to any such deduction problem—i.e., automatically construct the appropriate deduction, or proof—provided that the problem is solvable. The proviso sounds unnecessary: how indeed could we expect the algorithm to construct a proof as solution in cases where no solution exists? The intention, however, behind the proviso is to state that the algorithm will not, in general, give any indication that no solution is forthcoming when this is in fact the case; when applied to unsolvable deduction problems (namely, those in which $\mathcal{B}$ does not logically follow from the given assumptions $\mathcal{A}_1,\ldots,\mathcal{A}_n$), the calculation process called for by the algorithm will in general continue forever in theory and in practice be eventually abandoned as inconclusive. Thus the algorithm is capable of settling deduction problems only if they are solvable; it cannot in general inform us that a deduction does not exist. On these grounds there are those who would prefer to call the procedure a *semi-algorithm* rather than an *algorithm,* reserving the latter term for those calculation procedures which always terminate after a finite (though not necesasrily predictable) amount of calculation; but so long as the two possibilities are borne in mind, the nomenclature is not of any great importance.

Now it is not merely (as in the case of von Neumann's chess algorithm) that we have a proof of the existence of such a calculating procedure for the construction of proofs, without being able to exhibit and use it; we actually have the procedure itself. In fact, there is not just one procedure; there are many variants, which differ quite widely from each other. All share the property of allowing us to construct deductions automatically, in a machine-like manner; indeed the necessary calculations can be performed entirely by a computing machine, which need only be supplied in each case with the statement $\mathcal{B}$ and the assumptions $\mathcal{A}_1,\ldots,\mathcal{A}_n$ in a suitably encoded form, together of course with the complete set of instructions constituting the calculation procedure.

It is surely of immense philosophical interest, as well as being of great practical importance, that the task of deduction which is performed by human beings only through their use of intelligent ingenuity, of a subtle feeling for structure and abstract formal beauty, and of other sublime and

elusive characteristics of the human mind, should also be performable by the well-defined, analytical, step-by-step behavior of a purely mechanical deterministic automaton. At the present time, however, the development of automatic deduction procedures is in its infancy. One is reminded of Dr. Johnson's remark likening a woman's preaching to a dog's walking on its hind legs: it is not done well; the wonder is that it is done at all.

The purpose of the remainder of this article is to discuss the shortcomings of known automatic deduction procedures and to describe a recent theoretical development, by the author of this paper, which apparently eliminates entirely one of the major obstacles to their practical exploitation. We shall see that there still remain further problems to be solved before automatic deductions on computing machines will replace routinely and on a large scale the heuristic deductions of the highly-trained intelligent human; but research towards this end is still at a very early stage, pursued in relatively few centers by relatively few people. The outlook for the future is extremely promising, and the goal is exciting. It is no less than the ability to imitate and amplify our own intellectual powers by artificial means.

The Davis-Putnam Procedure

Martin Davis and Hilary Putnam have presented a calculating procedure for automatically demonstrating the unsatisfiability of any unsatisfiable, finite set of statements expressed in the artificial language known to logicians as the *first-order predicate calculus with function symbols.*[4] This language is known to be rich enough in expressive power for formulating all of extant mathematics, and is the natural instrument to use, in place of ordinary, informal language, for mathematical analysis of logical problems. Each statement of this language has two sorts of symbol; the *logical* symbols *(not, and, or, if . . . then, if and only if, there exists, for all)* which have a fixed meaning, and the *nonlogical* symbols (the *predicates, function symbols, individual constants,* and *individual variables*) which can be given different meanings or interpretations. Thus, more exactly, a statement in this language is really only a form or pattern or structure of a statement, namely that which is common to all those statements obtainable by interpreting the nonlogical symbols in all the possible ways. For example, the simple statement-pattern

$$(Ex)\ (A(x)\ \&\ B(x))$$

is read: *there exists an x such that x is an A and x is a B.* By interpreting the individual variable x and the predicates A and B in different ways, we see for instance that each of the following quite different statements are all of this one pattern or form:

there is an *integer* which is *even* and *prime*;

there is an *animal* which is *cloven-hoofed* and *feathered*;
there is a *set* which is *infinite* and *nondenumerable.*

Thus a statement-pattern, or *formula,* might turn into a true statement for some of its possible interpretations, and a false statement for the other possible interpretations. A formula is called *satisfiable* if there is at least one way of interpreting it so that it becomes a true statement; likewise a set of formulae is said to be satisfiable if there is at least one way of interpreting all the nonlogical symbols in the formulae so that each formula in the set becomes a true statement. A formula, or set of formulae, which is not satisfiable in this sense is said to be *unsatisfiable.* For example, the following set

$$\{(Ax)B(x),\ (Ey)\sim B(y)\}$$

of two formulae is unsatisfiable; for the first says that *every x is a B* while the second says that *there is a y which is not a B;* and there is no way of interpreting *'x,' 'y'* and *'B'* which will turn both formulae simultaneously into true statements.

There is an intimate connection between the notion of unsatisfiability of a set of formulae and the notion of one formula $\mathcal{B}$ being a consequence of, or following from, a set $\{\mathcal{A}_1,\ldots,\mathcal{A}_n\}$ of formulae. For $\mathcal{B}$ follows from $\{\mathcal{A}_1,\ldots,\mathcal{A}_n\}$ just in case the set $\{\mathcal{A}_1,\ldots,\mathcal{A}_n,\sim\mathcal{B}\}$ is unsatisfiable, i.e., just in case there is no interpretation which will turn each of $\mathcal{A}_1,\ldots,\mathcal{A}_n$ into a true statement while turning $\mathcal{B}$ into a false statement.

Thus Davis and Putnam's calculating procedure is easily used as an automatic deduction procedure; to deduce $\mathcal{B}$ from $\{\mathcal{A}_1,\ldots,\mathcal{A}_n\}$ we simply do the calculation to demonstrate the set $\{\mathcal{A}_1,\ldots,\mathcal{A}_n,\sim\mathcal{B}\}$ to be unsatisfiable.

In order to explain the essentials of the Davis-Putnam procedure without getting lost in a welter of logical details we shall state without proof that every set of formulae $\{\mathcal{A}_1,\ldots,\mathcal{A}_m\}$ can be transformed by a series of straightforward operations into another set of formulae $\{\mathcal{B}_1,\ldots,\mathcal{B}_n\}$ which has a very special standard form to make calculating easy, and, what is most important, which satisfies the condition:

$\{\mathcal{B}_1,\ldots,\mathcal{B}_n\}$ *is unsatisfiable if and only if* $\{\mathcal{A}_1,\ldots,\mathcal{A}_m\}$ *is.*

Without more ado we shall from this point onwards assume that we always have our set of formulae in this standard form, which we shall now describe.

The set $\{\mathcal{B}_1,\ldots,\mathcal{B}_n\}$ is in standard form just in case each of the $\mathcal{B}_j$ in the set is a *disjunction* of *literals*; a *literal* is either an *atomic formula* or else the *negation of an atomic formula*; an *atomic formula* is an expression consisting of a *predicate symbol* followed by one or more terms, separated by commas and enclosed in a pair of parentheses; finally a *term* is any expression which is either an *individual variable,* an *individual constant,* or else

consists of a *function symbol* followed by one or more terms, separated by commas and enclosed in a pair of parentheses.

We shall use capital italic letters for predicate symbols, lower case italic letters from the top of the alphabet ("*t*" onwards) for individual variables, lower case italic letters from the bottom of the alphabet ("*a*" to "*s*" inclusive) for individual constants and function symbols, with numerical subscripts if needed.

Since the order and multiplicity of literals in a disjunction are logically irrelevant, we shall treat each $\mathcal{B}_j$ in the set $\{\mathcal{B}_1,...,\mathcal{B}_n\}$ as simply the set of literals it contains; thus the pattern of the systems we deal with is the very simple one of *a finite set of finite sets of literals.* Davis and Putnam call a finite set of literals a *clause,* and we will also follow this usage. In order to complete our vocabulary of terse names, let us call a finite set of clauses a *structure.* The following are examples of structures:

$\mathcal{S}_1$: $\{\{B(x)\}, \{\sim B(a)\}\}$;

$\mathcal{S}_2$: $\{\{P(x,g(x),y,h(x,y),z,k(x,y,z))\}, \{\sim P(u,v,m(v),w,n(v,w),t)\}\}$;

$\mathcal{S}_3$: $\{\{\sim P(x,y,u),\sim P(y,z,v),\sim P(x,v,w),P(u,z,w)\},$
$\{\sim P(x,y,u),\sim P(y,z,v),\sim P(u,z,w),P(x,v,w)\},$
$\{P(g(x,y),x,y)\}, \{P(x,h(x,y),y)\}, \{P(x,y,f(x,y))\},$
$\{\sim P(k(x),x,k(x))\}\}$;

$\mathcal{S}_4$: $\{\{N(a)\}, \{\sim N(x),N(s(x))\}\}$.

The individual variables in each clause of a structure are tacitly understood to have the *universal* interpretation; each clause is thus interpreted to mean what it would normally mean if it were prefixed with universal quantifiers binding its variables, with disjunction symbols supplied between its literals, in place of the commas.

It is to *structures,* then, that the Davis-Putnam calculating procedure is to be applied. In order to explain the calculating procedure, we next introduce the concept of the *Herbrand Universe* of a structure.

Each structure has a *vocabulary of constants* $\mathcal{F}_0$, $\mathcal{F}_1$,...,$\mathcal{F}_k$; where $\mathcal{F}_i$ is the set of function symbols occurring in the structure with i terms as arguments, and $\mathcal{F}_0$ is the set of individual constants which occur in the structure (if *none* occur, as in $\mathcal{S}_2$ and $\mathcal{S}_3$ above, then $\mathcal{F}_0$ is the set $\{a\}$). The Herbrand Universe of the structure with vocabulary $\mathcal{F}_0$,...,$\mathcal{F}_k$ is then defined to be *the set of all terms constructible solely from the symbols in the vocabulary* $\mathcal{F}_0$,...,$\mathcal{F}_k$. This set $\mathcal{H}$ may be finite or infinite, depending on the structure's vocabulary. For example, the $\mathcal{H}$ for the structure $\mathcal{S}_1$ above is finite, being simply the set {a}; but $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$ each have infinite Herbrand Universes, that of $\mathcal{S}_4$ being, for example:

$$\{a, s(a), s(s(a)), s(s(s(a))),...,\}$$

while those of $\mathcal{S}_2$ and $\mathcal{S}_3$ have a more elaborate form.

One of the basic operations in the Davis-Putnam calculating procedure is called *instantiation,* and it consists of constructing a new clause from one of the clauses in the given structure by replacing each variable with a member of the structure's Herbrand Universe $\mathcal{H}$. The same member of $\mathcal{H}$ must replace each occurrence of the same variable, but different variables may be replaced by different members of $\mathcal{H}$. Any structure which consists entirely of clauses obtained in this way from clauses of the structure $\mathcal{S}$ is called a *substructure* of $\mathcal{S}$. It is easy to see that a structure $\mathcal{S}$ which has an infinite Herbrand Universe will have infinitely many distinct substructures, while an $\mathcal{S}$ with a finite Herbrand Universe has only a finite number of substructures. For example, the structure $\mathcal{S}_1$ has only the following three substructures:

$$\{\{B(a)\}\};$$
$$\{\{\sim B(a)\}\};$$
$$\{\{B(a)\}, \{\sim B(a)\}\}.$$

It will be noticed from our definition of substructure that none of the literals appearing in a substructure can contain any variables. They all represent basic statements which become true or false the moment an interpretation is given to the predicate symbols, individual constants, and function symbols which they contain. As a result of this, we can very much simplify the notion of an interpretation for substructures; essentially, from the abstract point of view, a substructure is given a meaning when we specify which of its literals is true and which is false. We need not specify their meaning over and above their truth value. Once an interpretation of a substructure is given in this way, the substructure is satisfied by it just in case there is no clause in the substructure which does not have at least one literal in it which is true in the interpretation. Collecting all these ideas and boiling them down to the bare essentials, it is not difficult to see that *a substructure is satisfiable if and only if there is a set $\mathcal{J}$ of its literals with the property that no clause in the substructure fails to contain at least one member of $\mathcal{J}$, and with the further property that $\mathcal{J}$ does not contain a pair of literals one of which is simply the negation of the other.*

Such a set $\mathcal{J}$ of literals is thought of as a specification of which literals are to be true in the tacitly understood interpretation. Now we can say simply that a substructure is unsatisfiable just in case it is not satisfiable in the above sense. The third substructure of $\mathcal{S}_1$, for instance, is unsatisfiable, while the first and second are both satisfiable.

Now the Davis-Putnam calculating procedure is based on the fundamental fact, discovered by Herbrand,[5] that *a structure $\mathcal{S}$ is unsatisfiable if and only if it has an unsatisfiable finite substructure.* This theoretical result is one of the most important and beautiful in all of modern logical theory; it reduces the question of the satisfiability of an arbitrary structure $\mathcal{S}$,

which involves the impossible task of "searching" the class of all possible interpretations of $\mathcal{S}$, to a series of answerable questions concerning the finite substructures of $\mathcal{S}$. One can always discover, in a finite number of steps, whether a finite substructure is satisfiable or not.

The specific calculation to be done on a structure $\mathcal{S}$ is organized around two principles: the first principle is a systematic way of setting out the finite substructures of $\mathcal{S}$ in a sequence

$$\mathcal{S}^1, \mathcal{S}^2, \mathcal{S}^3, \ldots,$$

of such a kind that every finite substructure of $\mathcal{S}$ will eventually occur at some place in the sequence. There are many ways of doing this, as we shall see. The second principle is simply a systematic way of testing any given finite substructure for satisfiability.

Again, there are a great many different ways of doing this. Historically, Davis and Putnam's calculation involved two specific choices for the first and second principles; but for the purposes of our present discussion we need not now enter into these. We shall instead consider any two specific choices for the first and second principles as *a Davis-Putnam procedure.* Our objective is for the moment to discuss the general case in abstraction from any particular exemplification of it.

One useful fact may be noted, in connection with the enumeration of finite substructures. The concept of substructure also applies to substructures themselves; one substructure $\mathcal{S}^i$ being a substructure of another one $\mathcal{S}^j$ just in case $\mathcal{S}^i \subseteq \mathcal{S}^j$ (i.e., all the clauses of $\mathcal{S}^i$ are clauses of $\mathcal{S}^j$). It is easy to see that if $\mathcal{S}^i$ is unsatisfiable, so is any $\mathcal{S}^j$ such that $\mathcal{S}^i \subseteq \mathcal{S}^j$; hence it would suffice to examine each of a sequence $\mathcal{S}^1, \mathcal{S}^2, \ldots,$ which has the property that every finite substructure is a subset of at least one $\mathcal{S}^i$ in the sequence. Davis and Putnam themselves exploited this principle. But however the enumeration of the finite substructures is done, there will always be an earliest point in the sequence when an unsatisfiable finite substructure is detected, if the method is applied to an unsatisfiable structure $\mathcal{S}$. Indeed, the matter can be put in this way: that each method of enumeration associates a number with each unsatisfiable structure $\mathcal{S}$, namely, *the earliest point in the enumeration of the substructures of* $\mathcal{S}$ *at which an unsatisfiable finite substructure occurs.* A second number is likewise associated, namely the *number of clauses* in this earliest unsatisfiable finite substructure of $\mathcal{S}$. If either (or both) of these two numbers is very large for a given structure $\mathcal{S}$ and given method of enumeration, then the calculation will not be practically possible for that $\mathcal{S}$ using that method.

The unfortunate fact about Davis-Putnam calculations is that, for almost all "interesting" structures $\mathcal{S}$, and for all the methods of enumeration that have been tried, *one or the other of these two numbers is absolutely enormous.* The practical consequence of this is that the calculations cannot

be done, even on the fastest and largest of modern electronic computers. This enormity problem can be easily illustrated with the help of the structures $\mathcal{S}_2$ and $\mathcal{S}_3$ exhibited earlier. The particular method of enumeration we use in this illustration is that used by Quine,[6] Dreben,[7] Gilmore[8] and is very much like that of Davis and Putnam. To follow it, we first note that any Herbrand Universe $\mathcal{H}$ falls naturally into finite subsets $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathcal{H}_2$,..., where in general $\mathcal{H}_j$ is just the *set of all terms in* $\mathcal{H}$ *the maximum depth of composition of which is j or less.* For example, for $\mathcal{S}_3$ these finite subsets of $\mathcal{H}$ look like:

$\mathcal{H}_0$: $\{a\}$
$\mathcal{H}_1$: $\{a, k(a), g(a,a), h(a,a), f(a,a)\}$
$\mathcal{H}_2$: $\{a, k(a), g(a,a), h(a,a), f(a,a), k(k(a)),$
$k(g(a,a))$,..., etc.$\}$

These subsets are sometimes called the *levels* of the Herbrand Universe. They give a kind of pattern to the otherwise confusing combinatorial jungle of $\mathcal{H}$. Now the substructure obtained from a structure $\mathcal{S}$ by forming *all* instances of clauses in $\mathcal{S}$ that can be formed using terms from a finite subset $\mathcal{G}$ of $\mathcal{H}$ is denoted by $\mathcal{G}(\mathcal{S})$, and is said to be *the* $\mathcal{G}$*-saturated substructure of* $\mathcal{S}$. It is of course finite, since $\mathcal{G}$ and $\mathcal{S}$ are finite. It is clear that if

$$\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2,\ldots,$$

is any sequence of finite subsets of $\mathcal{H}$ with the property that

$$\mathcal{H} = \bigcup_{j=0}^{\infty} \mathcal{G}_j,\ \mathcal{G}_j \subseteq \mathcal{G}_{j+1} \text{ for all } j,$$

then the sequence of $\mathcal{G}_j$-saturated substructures of $\mathcal{S}$,

$$\mathcal{G}_0(\mathcal{S}), \mathcal{G}_1(\mathcal{S}), \mathcal{G}_2(\mathcal{S}),\ldots,$$

will satisfy the conditions of the first principle of a Davis-Putnam calculation. But the successive *levels* of $\mathcal{H}$ have this property, i.e.,

$$\mathcal{H} = \bigcup_{j=0}^{\infty} \mathcal{H}_j,\ \mathcal{H}_j \subseteq \mathcal{H}_{j+1} \text{ for all } j;$$

and so the sequence of *level-saturated substructures*

$$\mathcal{H}_0(\mathcal{S}), \mathcal{H}_1(\mathcal{S}), \mathcal{H}_2(\mathcal{S}),\ldots,$$

is suitable for the first principle of a Davis-Putnam calculation. It is this sequence, which has a certain naturalness, that the authors cited have employed.

For the structure $\mathcal{S}_3$, the earliest value of j such that $\mathcal{H}_j(\mathcal{S}_3)$ is unsatisfiable is 3; and the number of clauses in $\mathcal{H}_3(\mathcal{S}_3)$ is approximately 10^{18}. For the structure $\mathcal{S}_2$, the earliest value of j such that $\mathcal{H}_j(\mathcal{S}_2)$ is unsatisfiable is 5; and the number of clauses in $\mathcal{H}_5$ $(\mathcal{S}_2)$ is approximately 10^{336}. These numbers are arrived at by quite simple combinatorial methods given

in another paper.[9] They are quite small examples of what one means by *enormous numbers.*

The method of enumerating finite substructures by means of saturation sequences $\mathcal{G}_0$, $\mathcal{G}_1$, $\mathcal{G}_2$,..., of finite subsets of $\mathcal{H}$ seems to be inherently prone to this difficulty. Yet there is a curious observation to be made: for any particular unsatisfiable structure $\mathcal{S}$ there is always a finite subset $\mathcal{P}$ of $\mathcal{H}$ which has the properties: $\mathcal{P}(\mathcal{S})$ is an unsatisfiable substructure; if $\mathcal{Q}$ is any proper subset of $\mathcal{P}$, $\mathcal{Q}(\mathcal{S})$ is a satisfiable substructure; and neither $\mathcal{P}$ nor $\mathcal{P}(\mathcal{S})$ are enormous. We have elsewhere[10] called such subsets $\mathcal{P}$ *proof sets.* For example, proof sets for $\mathcal{S}_2$ and $\mathcal{S}_3$ are:

for $\mathcal{S}_2$, $\mathcal{P} = \{a; g(a); m(g(a)); h(a,m(g(a)));$
$n(g(a),h(a,m(g(a)))); k(a,m(g(a)),n(g(a),h(a,m(g(a))))))\}$,
for $\mathcal{S}_3$, $\mathcal{P} = \{a, h(a,a), k(h(a,a)), g(k(h(a,a)),a)\}$

with 6 and 4 members respectively. $\mathcal{P}(\mathcal{S}_2)$ has 1512 clauses; $\mathcal{P}(\mathcal{S}_3)$ has roughly 10,000 clauses. Neither of these two numbers is enormous.

The claim made above that every unsatisfiable $\mathcal{S}$ has a *small* proof set $\mathcal{P}$ was too rash: we should have said, every $\mathcal{S}$ *for which a human could possibly discover a proof, or follow a proof, of unsatisfiability,* has a small proof set. With this qualification, the claim is quite defensible. For a proof which a human can grasp cannot itself be an *enormous* object. Indeed the sort of proof one normally encounters in mathematical contexts is quite short—a few pages at most, and most often much shorter than that. The number of distinct constructions or concepts in any one proof is also small—else the human mind could not handle the complexity with anything like the customary amount of effort and mental capacity that is available. These heuristic considerations, together with empirical evidence from a wide variety of proofs, have led us to suppose that the small size and limited complexity of "humanly possible" proof sets are both inherent properties of them, and reflect ultimately the limited capacity of the human mind to handle more than a certain amount of symbolic structural variety at any one time.

This same argument would lead to the conclusion that an unsatisfiable structure $\mathcal{S}$ whose proof set $\mathcal{P}$ is very large and complex may well be beyond the ability of the human to handle; even if a proof were discovered mechanically for such an $\mathcal{S}$, it might be too hard to follow—to keep in mind all at once.

Returning to the enormity problem evidenced by the saturation sequence method, we now see that, for each unsatisfiable $\mathcal{S}$, there *is* a saturation sequence which very quickly disposes of $\mathcal{S}$, namely, any sequence whose first subset is a proof set $\mathcal{P}$ for $\mathcal{S}$. What seems, however, to be the case is that there is no one saturation sequence which uniformly disposes this quickly of every $\mathcal{S}$. And that is precisely what a Davis-Putnam calculating

procedure must have if it is to be of any practical interest. This situation seems to be unavoidable, and has discouraged a number of people from pursuing further research on automatic deduction. If it were only possible to develop a calculating procedure which somehow simultaneously followed out all possible saturation sequences for a structure $\mathcal{S}$, there would be a glimmer of hope—for as we have seen, there is always an "optimal" saturation sequence for a given $\mathcal{S}$, the trouble being that the optimal saturation sequence is obtained by a different method for different structures. Such a uniform method would in effect pick out the proof set $\mathcal{P}$ for $\mathcal{S}$, much as humans do, or try to do, when attacking a proof-problem creatively.

From the discussion so far, one would suppose that such a method could not possibly exist. It is a pleasant surprise, therefore, to find that such a method does exist and has just the desirable properties described in the previous paragraph. We explain this new method in the next section.

The Resolution Method

We recall that a *saturation sequence* for a structure $\mathcal{S}$ is a sequence $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \ldots$, of finite subsets of the Herbrand Universe $\mathcal{H}$ of $\mathcal{S}$ which has the property

$$\mathcal{H} = \bigcup_{j=0}^{\infty} \mathcal{G}_j, \mathcal{G}_j \subseteq \mathcal{G}_{j+1} \text{ for all } j.$$

A Davis-Putnam procedure using this kind of sequence consists of testing successively the substructures

$$\mathcal{G}_0(\mathcal{S}), \mathcal{G}_1(\mathcal{S}), \mathcal{G}_2(\mathcal{S}), \ldots,$$

until one is found which is unsatisfiable. So far we have paid no attention to the methods by which a substructure can be tested for unsatisfiability, except to say that there are many of them. In order to lay the groundwork for the new automatic deduction procedure, we examine now a particular way of testing substructures for unsatisfiability which lends itself to theoretical investigation and to a certain generalization of which we shall make important use.

Suppose that we are given two clauses, $\mathcal{C}$ and $\mathcal{D}$, of some substructure. And suppose that one of the literals in $\mathcal{C}$ (say, $\mathcal{L}$) and one of the literals in $\mathcal{D}$ (say, $\mathcal{M}$) are exactly like each other except that one is negated and the other is not. Then the following new clause:

$$(\mathcal{C} - \{\mathcal{L}\}) \cup (\mathcal{D} - \{\mathcal{M}\})$$

constructed by pooling all the literals of $\mathcal{C}$ except $\mathcal{L}$ and all the literals of $\mathcal{D}$ except $\mathcal{M}$, is called a *resolvent* of the clauses $\mathcal{C}$ and $\mathcal{D}$. $\mathcal{C}$ and $\mathcal{D}$ may have several different resolvents, since there may be more than one pair $\mathcal{L}, \mathcal{M}$ of complementary literals with $\mathcal{L}$ and $\mathcal{M}$ in $\mathcal{C}$ and $\mathcal{D}$ respectively.

It is easy to see that if a substructure $\mathcal{A}$ is satisfiable, then so is any set $\mathcal{B}$ of clauses obtained by adding to $\mathcal{A}$ one or more resolvents of a pair of clauses in $\mathcal{A}$, and conversely. For if $\mathcal{A}$ is satisfiable, there is a set $\mathcal{J}$ of its literals such that every clause in $\mathcal{A}$ contains a member of $\mathcal{J}$, but no pair of *complementary* literals is in $\mathcal{J}$. Thus in forming a resolvent

$$(\mathcal{C} - \{\mathcal{L}\}) \cup (\mathcal{D} - \{\mathcal{M}\})$$

of two clauses $\mathcal{C}$ and $\mathcal{D}$ of the substructure $\mathcal{A}$, we form a clause which must also contain a literal in $\mathcal{J}$ since each of $\mathcal{C}$ and $\mathcal{D}$ do, and these cannot be both $\mathcal{L}$ and $\mathcal{M}$, which are complementary.

Since any such set $\mathcal{B}$ of clauses is satisfiable if and only if $\mathcal{A}$ is satisfiable, this is true in particular of the set obtained by adding to $\mathcal{A}$ all the resolvents of pairs of clauses in $\mathcal{A}$. This set is denoted by $\mathcal{R}(\mathcal{A})$, and is called the *resolution of* $\mathcal{A}$. We are to think of the resolution of any set $\mathcal{A}$ of clauses as another set $\mathcal{R}(\mathcal{A})$ obtained from $\mathcal{A}$ by the performance of a single operation on $\mathcal{A}$ (although of course it can be a rather complicated, long operation).

From the definition of *resolvent* it is easy to see that two clauses $\mathcal{C}$ and $\mathcal{D}$ might have *the empty clause* as a resolvent. This will happen if $\mathcal{C}$ is $\{\mathcal{L}\}$ and $\mathcal{D}$ is $\{\mathcal{M}\}$ and $\mathcal{L}$ and $\mathcal{M}$ are complementary literals. And any set of clauses which contains the empty clause { } is of course *unsatisfiable*; for there can be no set $\mathcal{J}$ of literals such that { } contains one of the literals in $\mathcal{J}$. Thus if $\mathcal{R}(\mathcal{A})$ contains { }, where $\mathcal{A}$ is any set of clauses, then $\mathcal{A}$ is unsatisfiable.

This observation leads to a smooth, uniform way of testing a substructure $\mathcal{A}$ for unsatisfiability. In order to depict it compactly, we introduce a generalization of the notation $\mathcal{R}(\mathcal{A})$ by the convention that

$$\mathcal{R}^0(\mathcal{A}) \text{ is } \mathcal{A},$$
$$\text{and} \qquad \mathcal{R}^{n+1}(\mathcal{A}) \text{ is } \mathcal{R}(\mathcal{R}^n(\mathcal{A})).$$

We now call the set $\mathcal{R}^n(\mathcal{A})$ the nth *resolution of* $\mathcal{A}$, so that $\mathcal{R}(\mathcal{A})$, *the resolution of* $\mathcal{A}$, is now also called the *first resolution of* $\mathcal{A}$.

Now if $\mathcal{A}$ is a finite substructure, there must be a point at which $\mathcal{R}^n(\mathcal{A})$ and $\mathcal{R}^{n+1}(\mathcal{A})$ are the same set of clauses. To see this, note that there can be only finitely many literals in $\mathcal{A}$, and hence only finitely many *sets* of these literals. Thus we cannot continue to add further sets of these literals without eventually exhausting all the possible ones. Any set $\mathcal{B}$ of clauses which has the property that

$$\mathcal{R}(\mathcal{B}) = \mathcal{B}$$

is called a *resolved* set of clauses, provided only that it does not contain { }. We can now see that for any finite substructure $\mathcal{A}$ the sequence

$$\mathcal{R}^0(\mathcal{A}), \mathcal{R}^1(\mathcal{A}), \mathcal{R}^2(\mathcal{A}), \ldots,$$

must eventually terminate in a resolved set of clauses or in a set of clauses

that contains { }. In the first case, $\mathcal{A}$ is satisfiable, in the second case, $\mathcal{A}$ is unsatisfiable.

In order to prove that $\mathcal{A}$ is satisfiable if one of its nth resolutions is a resolved set, we need only show that *any* resolved set $\mathcal{B}$ of clauses is satisfiable. And in order to show this, we need only indicate how to construct the required set $\mathcal{J}$ of literals. To this end, suppose that $\mathcal{L}_1, \mathcal{L}_2,\ldots,\mathcal{L}_k$ is a complete list of the atomic formulas which occur in the (finite) resolved set $\mathcal{B}$, whether negated or not; and let $\mathcal{M}_1, \mathcal{M}_2,\ldots,\mathcal{M}_k$ be the list of their respective negations. We then define the set $\mathcal{J}$ to comprise the literals on the list $\mathcal{J}_1, \mathcal{J}_2,\ldots,\mathcal{J}_k$, where we choose the successive literals $\mathcal{J}_j$ for $j = 1,\ldots,k$ by the following method: $\mathcal{J}_j$ is to be $\mathcal{L}_j$, unless some clause in $\mathcal{B}$ consists entirely of complements of literals on the partial list $\mathcal{J}_1,\ldots, \mathcal{J}_j$ thereby determined; in that case $\mathcal{J}_j$ is to be $\mathcal{M}_j$. It is clear that the set $\mathcal{J}$ so determined does not contain a pair of complementary literals; but also every clause in $\mathcal{B}$ contains at least one literal in $\mathcal{J}$. For suppose not; then for some smallest number j, $1 \leq j \leq k$, there must be a clause $\mathcal{C}$ in $\mathcal{B}$ which consists entirely of complements of literals in the partial list $\mathcal{J}_1,\ldots,\mathcal{J}_j$. By the method of choosing the elements of $\mathcal{J}$, $\mathcal{J}_j$ must therefore be $\mathcal{M}_j$. Since j is the smallest such number, the clause $\mathcal{C}$ must contain $\mathcal{L}_j$. But since $\mathcal{J}_j$ was chosen to be $\mathcal{M}_j$, there must exist in $\mathcal{B}$ a clause $\mathcal{D}$ which consists entirely of complements of literals in the list $\mathcal{J}_1,\ldots, \mathcal{J}_{j-1}, \mathcal{L}_j$. (Note that $j > 1$ because $\mathcal{B}$ is a resolved set and cannot contain both of the clauses $\{\mathcal{L}_j\}$ and $\{\mathcal{M}_j\}$, for then it would have to contain { }). Again, since j is the smallest such number, $\mathcal{D}$ must contain $\mathcal{M}_j$. But then the clause

$$(\mathcal{C} - \{\mathcal{L}_j\}) \cup (\mathcal{D} - \{\mathcal{M}_j\})$$

consists entirely of complements of literals in the partial list $\mathcal{J}_1,\ldots,\mathcal{J}_{j-1}$. Since this clause is a resolvent of $\mathcal{C}$ and $\mathcal{D}$, and since $\mathcal{B}$ is a resolved set of clauses, we conclude that this clause is in $\mathcal{B}$, and thus that the leastness of the number j is contradicted. Hence every clause in $\mathcal{B}$ contains a literal in $\mathcal{J}$, and $\mathcal{B}$ is satisfiable.

The outcome of this theoretical discussion is that in the procedure of calculating successively the nth resolutions of a substructure $\mathcal{B}$:

$$\mathcal{R}^0(\mathcal{B}), \mathcal{R}^1(\mathcal{B}), \mathcal{R}^2(\mathcal{B}),\ldots,$$

we have a complete test for the unsatisfiability of $\mathcal{B}$: $\mathcal{B}$ is unsatisfiable if *and only if* $\mathcal{R}^n(\mathcal{B})$ contains the empty clause { }, where n is the smallest number for which $\mathcal{R}^n(\mathcal{B}) = \mathcal{R}^{n+1}(\mathcal{B})$. Such a number is guaranteed to exist by the finiteness of the substructure $\mathcal{B}$.

Now we can express very compactly, using this *resolution method,* the overall calculating procedure of the Davis-Putnam type which uses the saturation sequence technique for generating successive substructures of the structure $\mathcal{S}$. As before, let $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2,\ldots$ be any sequence of finite subsets of the Herbrand Universe $\mathcal{H}$ of $\mathcal{S}$ with the property

$$\mathcal{H} = \bigcup_{j=0}^{\infty} \mathcal{G}_j,\ \mathcal{G}_j \subseteq \mathcal{G}_{j+1} \text{ for all } j.$$

Then the structure $\mathcal{S}$ is unsatisfiable if and only if, for some j and some n, the set

$$\mathcal{R}^n(\mathcal{G}_j(\mathcal{S}))$$

contains the empty clause.

We are now in a position to introduce the key idea of the new automatic deduction procedure, which consists essentially of a certain generalization of the concept of *resolution*. The operation as we have defined it so far applies only to *substructures* and their nth resolutions, i.e., to finite sets of clauses none of whose literals contain any variables. We shall extend the operation of resolution so that it also applies to *structures* in general, the new feature being that the clauses in structures can contain literals which have variables in them.

When we earlier introduced the notion of *instantiation* we discussed it only very briefly. It now becomes necessary to analyze this operation fully and carefully, with the help of an appropriate notation.

Let $\mathcal{C}$ be any clause, $\mathcal{V}_1,...,\mathcal{V}_n$ any set of distinct variables, and $\mathcal{T}_1,...,\mathcal{T}_n$ any terms. Then by

$$\mathcal{C}[\mathcal{T}_1/\mathcal{V}_1,..., \mathcal{T}_n/\mathcal{V}_n]$$

we denote that clause which consists of all the distinct literals

$$\mathcal{L}[\mathcal{T}_1/\mathcal{V}_1,..., \mathcal{T}_n/\mathcal{V}_n]$$

where $\mathcal{L}$ is in $\mathcal{C}$, and $\mathcal{L}[\mathcal{T}_1/\mathcal{V}_1,..., \mathcal{T}_n/\mathcal{V}_n]$ is the literal obtained from $\mathcal{L}$ by rewriting $\mathcal{L}$ from left to right, but writing $\mathcal{T}_i$ in the new literal wherever $\mathcal{V}_i$ occurs in $\mathcal{L}$. For example:

$$P(x,y,u)\ [g(x,y)/x,\ x/y,\ y/u,\ f(x,y)/z]$$

is the literal

$$P(g(x,y),x,y)$$

and

$$\{\sim P(x,y,u)\ \sim P(y,z,v)\ \sim P(x,v,w)\ P(u,z,w)\}\ [y/v,\ u/w]$$

is

$$\{\sim P(x,y,u)\ \sim P(y,z,y)\ P(u,z,u)\}.$$

Whenever the terms $\mathcal{T}_1,...,\mathcal{T}_n$ are all in the Herbrand Universe $\mathcal{H}$ of a structure $\mathcal{S}$, $\mathcal{C}$ is a clause in $\mathcal{S}$, and $\mathcal{V}_1,...,\mathcal{V}_n$ are all the distinct variables which occur in $\mathcal{C}$, then the clause $\mathcal{C}[\mathcal{T}_1/\mathcal{V}_1,...,\mathcal{T}_n/\mathcal{V}_n]$ is precisely what we earlier defined to be an *instance* of $\mathcal{C}$, and the substructures of $\mathcal{S}$ are thus finite sets of clauses each having the form

$$\mathcal{C}[\mathcal{T}_1/\mathcal{V}_1,...,\mathcal{T}_n/\mathcal{V}_n]$$

with $\mathcal{C}$ in $\mathcal{S}$, $\mathcal{V}_1,...,\mathcal{V}_n$ all the distinct variables of $\mathcal{C}$, and $\mathcal{T}_1,...,\mathcal{T}_n$ in $\mathcal{H}$.

Consider then the substructure $\mathcal{G}(\mathcal{S})$, where $\mathcal{G}$ is any finite subset of

the Herbrand Universe $\mathcal{H}$ of $\mathcal{S}$. From the above remark, every clause in $\mathcal{G}(\mathcal{S})$ has the form

$$\mathcal{C}[\mathcal{T}_1/\mathcal{V}_1,\ldots,\mathcal{T}_n/\mathcal{V}_n]$$

for some $\mathcal{C}$ in $\mathcal{S}$; $\mathcal{V}_1,\ldots,\mathcal{V}_n$ being all the distinct variables of $\mathcal{C}$ and $\mathcal{T}_1,\ldots,\mathcal{T}_n$ being terms in the set $\mathcal{G}$. By the definition of $\mathcal{G}(\mathcal{S})$, the converse is true—any such clause is a member of the substructure $\mathcal{G}(\mathcal{S})$.

Next, consider the resolution $\mathcal{R}(\mathcal{G}(\mathcal{S}))$ of this substructure. Any clause $\mathcal{K}$ in $\mathcal{R}(\mathcal{G}(\mathcal{S}))$ which is not already in $\mathcal{G}(\mathcal{S})$ must be a resolvent of two clauses which are already in $\mathcal{G}(\mathcal{S})$. Let these two clauses be

$$\mathcal{C}[\mathcal{T}_1/\mathcal{V}_1,\ldots,\mathcal{T}_n/\mathcal{V}_n] \text{ and } \mathcal{D}[\mathcal{U}_1/\mathcal{W}_1,\ldots,\mathcal{U}_r/\mathcal{W}_r]$$

where $\mathcal{C}$ and $\mathcal{D}$ are clauses in $\mathcal{S}$, $\mathcal{T}_1,\ldots,\mathcal{T}_n,\mathcal{U}_1,\ldots,\mathcal{U}_r$ are in $\mathcal{G}$, $\mathcal{V}_1,\ldots,\mathcal{V}_n$ are all the distinct variables in $\mathcal{C}$, and $\mathcal{W}_1,\ldots,\mathcal{W}_r$ are all the distinct variables in $\mathcal{D}$.

Since $\mathcal{K}$ is a resolvent of $\mathcal{C}[\mathcal{T}_1/\mathcal{V}_1,\ldots,\mathcal{T}_n/\mathcal{V}_n]$ and $\mathcal{D}[\mathcal{U}_1/\mathcal{W}_1,\ldots,\mathcal{U}_r/\mathcal{W}_r]$, there must be a pair $\mathcal{L}$, $\mathcal{M}$ of complementary literals, such that $\mathcal{L}$ is in the clause $\mathcal{C}[\mathcal{T}_1/\mathcal{V}_1,\ldots,\mathcal{T}_n/\mathcal{V}_n]$, and $\mathcal{M}$ is in the clause $\mathcal{D}[\mathcal{U}_1/\mathcal{W}_1,\ldots,\mathcal{U}_r/\mathcal{W}_r]$.

There must therefore be one or more literals $\mathcal{L}_1,\ldots,\mathcal{L}_k$ in $\mathcal{C}$ which become the literal $\mathcal{L}$ under the substitution $[\mathcal{T}_1/\mathcal{V}_1,\ldots,\mathcal{T}_n/\mathcal{V}_n]$, and one or more literals $\mathcal{M}_1,\ldots,\mathcal{M}_p$ in $\mathcal{D}$ which become the literal $\mathcal{M}$ under the substitution $[\mathcal{U}_1/\mathcal{W}_1,\ldots,\mathcal{U}_r/\mathcal{W}_r]$.

Suppose, then, in view of all this, that we ask the following question concerning the clauses $\mathcal{C}$ and $\mathcal{D}$: what are all of the ways in which such a clause $\mathcal{K}$ can be obtained, first by substituting terms from $\mathcal{G}$ for the variables of $\mathcal{C}$ and $\mathcal{D}$ so as obtain two clauses which contain one each of a pair of complementary literals, and then resolving the two clauses so obtained? It turns out that there are only finitely many such ways (perhaps, in some cases, none at all), and that they can all be represented in the form of the substructure

$$\mathcal{G}(\mathcal{B})$$

of a certain structure $\mathcal{B}$ which is computed directly from $\mathcal{C}$ and $\mathcal{D}$. Reserving the description of the computation for the Appendix, we state here that the structure $\mathcal{B}$ so computed is either empty or else consists of a finite number $\mathcal{B}_1,\ldots,\mathcal{B}_k$ of clauses which have the property that any clause $\mathcal{K}$ as described above is in one of the sets

$$\mathcal{G}(\mathcal{B}_1),\ldots,\mathcal{G}(\mathcal{B}_k)$$

of clauses. These clauses $\mathcal{B}_1,\ldots,\mathcal{B}_k$ are the (generalized) resolvents of $\mathcal{C}$ and $\mathcal{D}$, and coincide with the resolvents as previously defined whenever $\mathcal{C}$ and $\mathcal{D}$ contain no variables.

In terms of the notation already introduced for the resolution of a set of clauses, we can express the situation described above by writing:

$$\mathcal{R}(\mathcal{G}(\mathcal{S})) \subseteq \mathcal{G}(\mathcal{R}(\mathcal{S})).$$

Here, $\mathcal{R}(\mathcal{S})$ now means the structure obtained by adding to $\mathcal{S}$ all the clauses that are resolvents of a pair $\mathcal{C},\mathcal{D}$ of clauses in $\mathcal{S}$ in the new, generalized sense. And we have stated that any resolvent of a pair of clauses in the $\mathcal{G}$-saturated substructure $\mathcal{G}(\mathcal{S})$ of $\mathcal{S}$ is a clause in the $\mathcal{G}$-saturated substructure of the resolution $\mathcal{R}(\mathcal{S})$ of $\mathcal{S}$.

Another way of putting the point is to say simply that the operations of $\mathcal{G}$-saturation and resolution *semicommute.*

It is a straightforward consequence of this basic property that

$$\mathcal{R}^n(\mathcal{G}(\mathcal{S})) \subseteq \mathcal{G}(\mathcal{R}^n(\mathcal{S}))$$

holds for any structure $\mathcal{S}$, any finite subset $\mathcal{G}$ of the Herbrand Universe $\mathcal{H}$ of $\mathcal{S}$ (indeed, any subset $\mathcal{G}$ whatever), and any natural number n. This relation is the fundamental theoretical result on which the new automatic deduction method is based; we are now in a position to state the new method.

Recall that, where $\mathcal{G}_0$, $\mathcal{G}_1$, $\mathcal{G}_2$ is any saturation sequence for the structure $\mathcal{S}$, $\mathcal{S}$ is unsatisfiable if and only if the set

$$\mathcal{R}^n(\mathcal{G}_j(\mathcal{S}))$$

contains { } for some j and some n. Then by the fundamental property given above, we must also have that $\mathcal{S}$ is unsatisfiable if and only if the set

$$\mathcal{G}_j(\mathcal{R}^n(\mathcal{S}))$$

contains { } for some j and some n. But the $\mathcal{G}_j$-saturated substructure $\mathcal{G}_j(\mathcal{R}^n(\mathcal{S}))$ of the structure $\mathcal{R}^n(\mathcal{S})$ contains { } if and only if the structure $\mathcal{R}^n(\mathcal{S})$ contains { }. For mere instantiation over $\mathcal{G}_j$ cannot produce the empty clause { } from a nonempty clause. We conclude therefore that:

the structure $\mathcal{S}$ is unsatisfiable if and only if the structure $\mathcal{R}^n(\mathcal{S})$ contains { }, for some n.

This is the essential form of the new method of automatic deduction. The important feature of it is that the use of the saturation sequence $\mathcal{G}_0$, $\mathcal{G}_1$, $\mathcal{G}_2$,..., has dropped out. This means that the value of n for which $\mathcal{R}^n(\mathcal{S})$ first contains { } will be the same as it would have been for the *optimal* choice of saturation sequence for $\mathcal{S}$ using a Davis-Putnam method; but as we saw, the optimal saturation sequence would be one which *began* with a proof set $\mathcal{P}$ for $\mathcal{S}$. Hence the new method *implicitly* selects a proof set $\mathcal{P}$ for each different unsatisfiable structure $\mathcal{S}$ to which it is applied, but does so in a uniform way.

It is of interest that the structures $\mathcal{S}_2$ and $\mathcal{S}_3$ used as examples, in the previous section, of the enormity problem, behave with extreme docility under the new method. In fact, we have:

$\mathcal{R}^1(\mathcal{S}_2)$ contains { }
$\mathcal{R}^3(\mathcal{S}_3)$ contains { }.

In the next and final section we discuss the automatic deduction problem in the light of our resolution method.

The Future of Automatic Deduction

Although the resolution method of automatic deduction appears to solve the enormity obstacle which plagues Davis-Putnam methods, it is not clear that it does not have an enormity obstacle of its own. The parameter n in the method—the nth resolution of $\mathcal{S}$ being the earliest one to contain $\{\ \}$—can still be enormous, for some unsatisfiable $\mathcal{S}$; but worse, it is quite possible that the earliest $\mathcal{R}^n(\mathcal{S})$ to contain $\{\ \}$, even for small n, may often be an enormous structure, even though $\mathcal{S}$ is not. (Presumably we would never consider working with initial structures $\mathcal{S}$ which are themselves enormous.) In that case, the new method would be of no practical use for demonstrating the unsatisfiability of such structures $\mathcal{S}$.

We do not at present have sufficient theoretical knowledge concerning the rate of growth of $\mathcal{R}^n(\mathcal{S})$ with n and $\mathcal{S}$ to make a useful judgment in this matter. There is also lacking a sufficient body of empirical knowledge, derived from applying the resolution method to a wide variety of structures, on which we might base an evaluation. Both of these areas remain to be explored in future research.

There is a school of investigators who have been trying a somewhat different approach to attaining automatic deduction. This is the "heuristic" school, pioneered by Newell and Simon.[11] The method they adopt is essentially that of analyzing the strategies, maneuvers, and "tricks" of the skilled human practitioner of deduction, and executing these mechanically by suitably programming a computer. Analogous methods have been tried for complex board games such as checkers.[12] The attraction in this approach is that by its very nature it tends to bring understanding of human intellectual processes and to lead towards artificial reproductions of them in machine behavior. This is a very important objective of behavioral science and is rightly receiving more and more attention at the present time. Nevertheless, in the field of automatic deduction the successes of heuristic methods have been meager, when judged by the kinds of deduction which have been achieved and the effort required to achieve them. It does not seem likely that heuristic methods can possibly surpass the methods we have discussed in the body of this article; for these logical methods are based on a direct confrontation and study of the deduction problems themselves, construed as well-defined problems of combinatorial mathematics; whereas heuristic methods are intrinsically formulated at second remove from the deduction problems themselves, focussed as they are on the response of the human intelligence to these problems. Indeed, the response of a human to deduction problems is likely to be affected rather strongly by that human's knowl-

edge and understanding of their logical properties and of the efficacy of some of the hard-won combinatorial techniques for solving them. Furthermore the heuristic analysis of human problem-solving behavior is without much point unless it be directed at the most efficient and successful examples of such behavior. It may well be that before long the most efficient and successful problem-solving behavior—as far as deduction problems are concerned—will be that exhibited by machines in the process of executing precisely the sort of combinatorial calculations we have been discussing, or even by humans executing these same procedures, albeit at a slower pace than machines.

There is no avoiding the profound philosophical issue which lurks behind this entire discussion. What do we mean when we say that an intelligent human reasons deductively, or uses creative insight into a deduction problem in order to find a proof? It is difficult to find anything more in this kind of assertion than that a proof is found, and we cannot say how it is found. There appears to be an element of spontaneity, or randomness, or nonreproducibility in the human search procedure, as though the mind were roving through the possibilities in a not-completely-determined way, which it would be absurd to try to model by a systematic or algorithmic process. This may well be so. But there is no need to argue that human intellectual processes are automatic, in order to compare their performance with that of the automatic deduction procedures we have been discussing; nor can it be argued that since human intellectual processes are not automatic, they must necessarily be superior in performance to any process that is automatic. Indeed, that is just what remains to be seen. There is nothing intrinsically impossible in the idea that a human-designed artifact should outperform its human designers in the function that it is constructed to fulfil; we are already outlifted by the crane, outsped by the automobile, outleaped by the airplane and rocket, and outcomputed by the multiplying and dividing of the simple desk-model calculating machines.

It is not a foregone conclusion, at present, that we are going to be outclassed at solving deduction problems by suitably programmed automatic computing devices; but the odds are that we shall be, and overwhelmingly so. If and when this comes to pass, we must all be glad of it, for then there will be so much more time for us to think and to dream.

APPENDIX. THE CALCULATION OF RESOLVENTS.

A. The Factors of a Clause.

Let $\mathcal{C}$ be any clause $\{\mathcal{L}_1,\ldots,\mathcal{L}_k\}$. For each pair $\mathcal{L}_i$, $\mathcal{L}_j$ of distinct literals in $\mathcal{C}$, construct if possible a new clause from $\mathcal{C}$ by the following process:

Step 1. Set $\mathcal{C}^{(0)} = \mathcal{C}$, and $k = 0$. Then go to Step 2.

Step 2. Given $\mathcal{C}^{(k)}$ check whether $\mathcal{L}_i^{(k)} = \mathcal{L}_j^{(k)}$. If so, *terminate the process with* $\mathcal{C}^{(k)}$ *as the constructed clause.* Otherwise go to Step 3.

Step 3. Let $\mathcal{U}$ and $\mathcal{V}$ be the two symbols at which $\mathcal{L}_i^{(k)}$ and $\mathcal{L}_j^{(k)}$ first differ, reckoning from the left to the right. If neither of $\mathcal{U},\mathcal{V}$ is an individual variable, or else if one of $\mathcal{U},\mathcal{V}$ is an individual variable and the other is the initial symbol of a term containing that variable, then *terminate the process with no clause having been constructed.* Otherwise go to Step 4.

Step 4. If both of the symbols $\mathcal{U},\mathcal{V}$ are individual variables, let $\mathcal{T}$ be the earlier, and $\mathcal{W}$ the later of them, in the alphabetical order

$$t, u, v, w, x, y, z, t_1, u_1, v_1, w_1, x_1, y_1, z_1, t_2, \ldots$$

of the individual variables; while if one of $\mathcal{U},\mathcal{V}$ is an individual variable and the other is the initial symbol of a term, let $\mathcal{T}$ be that term and $\mathcal{W}$ be that variable; and in either case construct

$$\mathcal{C}^{(k+1)} = \mathcal{C}^{(k)}[\mathcal{T}/\mathcal{W}],$$

add 1 to k, and return to Step 2.

When applied to a clause $\mathcal{C}$ for all possible pairs of distinct literals in $\mathcal{C}$, this process yields zero or more clauses, which are called *immediate factors* of $\mathcal{C}$. The process is then applied to each of these clauses in turn, to obtain their immediate factors (if any). We continue in this way until no more clauses are produced. The set of all distinct clauses obtained by this iterative procedure, together with the clause $\mathcal{C}$ itself, constitutes the set of *factors* of the clause $\mathcal{C}$. Two clauses are not considered distinct if they differ only by a one-to-one change of variables.

The entire process described is called the *factorization algorithm,* and always yields a nonempty set of clauses when applied to a clause $\mathcal{C}$. (For at least $\mathcal{C}$ itself is in the set.)

B. The Resolvents of Two Clauses.

Let $\mathcal{C}$ and $\mathcal{D}$ be any two clauses. In order to compute all the *resolvents* (if any) of $\mathcal{C}$ and $\mathcal{D}$, execute the following procedure for each pair of *factors* $\mathcal{A},\mathcal{B}$ of $\mathcal{C},\mathcal{D}$ respectively, and each pair of literals $\mathcal{L}$ in $\mathcal{A}$ and $\mathcal{M}$ in $\mathcal{B}$:

Step 1. Check to see if one of $\mathcal{L},\mathcal{M}$ is negated and the other unnegated. If not, *terminate the procedure with no clause having been constructed.* Otherwise go to Step 2.

Step 2. Put $\mathcal{A}^{(0)} = \mathcal{A}[x_1/\mathcal{V}_1,\ldots,x_m/\mathcal{V}_m]$, $\mathcal{B}^{(0)} = \mathcal{B}[y_1/\mathcal{W}_1,\ldots,y_n/\mathcal{W}_n]$, and $k = 0$; where $\mathcal{V}_1,\ldots,\mathcal{V}_m$ are all the distinct variables of $\mathcal{A}$ in alphabetical order, and $\mathcal{W}_1,\ldots,\mathcal{W}_n$ are all the distinct variables of $\mathcal{B}$ in alphabetical order. Then go to Step 3.

Step 3. If $\mathcal{L}^{(k)}$ and $\mathcal{M}^{(k)}$ are identical except for the negation symbol, *terminate the procedure having constructed the following clause:*

$$(\mathcal{A}^{(k)} - \{\mathcal{L}^{(k)}\}) \cup (\mathcal{B}^{(k)} - \{\mathcal{M}^{(k)}\}).$$

Otherwise go to Step 4.

Step 4. Ignoring the negation symbol which precedes one of $\mathcal{L}^{(k)},\mathcal{M}^{(k)}$, let $\mathcal{U}$ and $\mathcal{V}$ be the two symbols with which $\mathcal{L}^{(k)}$ and $\mathcal{M}^{(k)}$ first differ. If neither of $\mathcal{U},\mathcal{V}$ is an individual variable, or else if one of them is an individual variable and the other is the initial symbol of a term containing that variable, *terminate the procedure with no clause having been constructed.* Otherwise go to Step 5.

Step 5. If both $\mathcal{U}$ and $\mathcal{V}$ are individual variables, let $\mathcal{T}$ be the earlier and $\mathcal{W}$ the later of them in alphabetical order; while if one of $\mathcal{U},\mathcal{V}$ is an individual variable and the other is the initial symbol of a term, let $\mathcal{T}$ be that term and $\mathcal{W}$ that variable; and in either case, construct

$$\mathcal{A}^{(k+1)} = \mathcal{A}^{(k)}[\mathcal{T}/\mathcal{W}],\ \mathcal{B}^{(k+1)} = \mathcal{B}^{(k)}[\mathcal{T}/\mathcal{W}],$$

add 1 to k, and return to Step 3.

The entire process described is called the *resolvent algorithm*; when this process is applied to a pair $\mathcal{C},\mathcal{D}$ of clauses, the set (possibly empty) of all the distinct clauses constructed by it is the set of *resolvents* of $\mathcal{C}$ and $\mathcal{D}$.

C. Remarks on the Two Algorithms.

As we observed in the body of the article, the resolvents (if any) computed by the resolvent algorithm for two clauses $\mathcal{C}$ and $\mathcal{D}$ *which contain no variables* (and which hence have no factors other than themselves) are simply the clauses (if any)

$$(\mathcal{C} - \{\mathcal{L}\}) \cup (\mathcal{D} - \{\mathcal{M}\})$$

where $\mathcal{L}$ is in $\mathcal{C}$, $\mathcal{M}$ is in $\mathcal{D}$, and $\mathcal{L}$ and $\mathcal{M}$ are complementary literals. This is quite easy to see from the way the resolvent algorithm is stated.

The proof that

$$\mathcal{R}(\mathcal{G}(\mathcal{S})) \subseteq \mathcal{G}(\mathcal{R}(\mathcal{S}))$$

(where $\mathcal{S}$ is any structure, $\mathcal{G}$ is any subset of the Herbrand Universe $\mathcal{H}$ of $\mathcal{S}$, and $\mathcal{R}(\mathcal{A})$, for any set $\mathcal{A}$ of clauses, is the set $\mathcal{A}$ together with all resolvents of all pairs of clauses in $\mathcal{A}$), is long and somewhat tedious. It is given in full in another paper[13] and we forego reproducing it here.

The notion of *factor,* and the factorization algorithm, are strictly auxiliary to the notion of *resolvent* and to the resolvent algorithm. It would have been possible to avoid their use altogether, at the cost of complicating the resolvent algorithm itself to an intolerable extent. As it is, one is slightly

uncomfortable at having to *define* a resolvent as something obtained by the performance of an elaborate piece of data-processing; but there seems to be no other way of doing it.

NOTES

1. Alonzo Church, "A Note on the Entscheidungsproblem," *Journal of Symbolic Logic*, Vol. 1, 1936, pp. 40-41. Correction, *ibid.*, pp. 101-102.
2. J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton, New Jersey, Princeton University Press, 1953.
3. *Ibid.*, p. 125.
4. M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *Journal of the Association for Computing Machinery*, Vol. 7, 1960, pp. 201-215.
5. J. Herbrand, "Recherches sur la théorie de la démonstration," *Travaux de la Société des Sciences et des Lettres de Varsovie, Classe III science mathématiques et physiques*, No. 33, 1930.
6. W. V. Quine, "A Proof Procedure for Quantification Theory," *Journal of Symbolic Logic*, Vol. 20, 1955, pp. 141-149.
7. B. Dreben, "On the Completeness of Quantification Theory," *Proceedings of the U. S. National Academy of Sciences*, Vol. 38, 1952, pp. 1047-1052.
8. P. C. Gilmore, "A Proof Method for Quantification Theory," *IBM Journal of Research and Development*, Vol. 4, 1960, pp. 28-35.
9. J. A. Robinson, "Theorem-proving on the Computer," *Journal of the Association for Computing Machinery*, Vol. 10, 1963, pp. 163-174.
10. *Ibid.*
11. A. Newell and H. A. Simon, "The Logic Theory Machine," *IRE Transactions on Information Theory*, Vol. IT-2, 1956, pp. 61-79.
12. A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, Vol. 3, 1959, pp. 210-229.
13. J. A. Robinson, "A Machine-oriented Logic Based on the Resolution Principle," to be published in a forthcoming issue of the *Journal of the Association for Computing Machinery*.